The Discrete Basis Problem

Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila

Abstract—Matrix decomposition methods represent a data matrix as a product of two factor matrices: one containing basis vectors that represent meaningful concepts in the data and another describing how the observed data can be expressed as combinations of the basis vectors. Decomposition methods have been studied extensively, but many methods return real-valued matrices. Interpreting real-valued factor matrices is hard if the original data is Boolean. In this paper, we describe a matrix decomposition formulation for Boolean data, the Discrete Basis Problem. The problem seeks for a Boolean decomposition of a binary matrix, thus allowing the user to easily interpret the basis vectors. We also describe a variation of the problem, the Discrete Basis Partitioning Problem. We show that both problems are NP-hard. For the Discrete Basis Problem, we give a simple greedy algorithm for solving it; for the Discrete Basis Partitioning Problem, we show how it can be solved using existing methods. We present experimental results for the greedy algorithm and compare it against other well-known methods. Our algorithm gives intuitive basis vectors, but its reconstruction error is usually larger than with the real-valued methods. We discuss the reasons for this behavior.

Index Terms—Mining methods and algorithms, clustering, classification, and association rules, text mining.

1 Introduction

Given an $n \times m$ matrix $\mathbf{C}$ and an integer $k < m$, classical matrix decomposition methods aim at finding an $n \times k$ matrix $\mathbf{S}$ and a $k \times m$ matrix $\mathbf{B}$ such that $\mathbf{C}$ can be approximately represented as the product of $\mathbf{S}$ and $\mathbf{B}$. The decomposition method represents the data by using k components: The matrix $\mathbf{B}$ tells how the components are related to the original attributes (columns), and the matrix $\mathbf{S}$ indicates how strongly each component is related to each row.

Singular value decomposition (SVD) [2] and nonnegative matrix factorization (NMF) [3] are typical examples of decomposition methods; the difference between the two is that NMF assumes nonnegative $\mathbf{C}$ and requires that $\mathbf{S}$ and $\mathbf{B}$ are nonnegative. Other matrix decomposition methods include latent Dirichlet allocation (LDA) [4] and multinomial PCA [5]; see Section 3 for additional discussion of related work.

These (and other) matrix decomposition methods allow the matrices $\mathbf{S}$ and $\mathbf{B}$ to contain arbitrary real numbers. However, if the input matrix $\mathbf{C}$ is binary, it is natural to require that $\mathbf{S}$ and $\mathbf{B}$ are also binary. In this paper, we consider the matrix decomposition problem created by this requirement. In this case, the combination operation of matrices $\mathbf{S}$ and $\mathbf{B}$ is the Boolean matrix product (i.e., the matrix product in the semiring of Boolean $\wedge$ and $\vee$).

The intuition behind using Boolean operations in the matrix decompositions is that sometimes the counts of the objects do not matter. For example, consider a course enrollment data set in a CS Department. Such a data set indicates which students enroll to which courses. Naturally, courses are divided into specialization areas. A student X interested in the Systems specialization needs to take, among others, courses on Operating Systems and Programming languages, and a student Y interested in the Software specialization needs to take courses on Compilers and Programming languages. On the other hand, a student Z interested in combining both of the above two specializations should take all courses: Compilers, Operating systems, and Programming languages (among others). Clearly, the student Z should be accounted for taking Programming languages only once. Thus, the set union operation is more appropriate for describing the actual data from the basis vectors (specialization areas).

Following the intuition in the previous example, we formulate the problem of finding a decomposition into binary matrices that give the best approximation to the input matrix. We call this problem the *Discrete Basis Problem* (DBP).

We show that DBP is NP-hard and it cannot be approximated unless $\mathrm{P} = \mathrm{NP}$. We give a simple greedy algorithm for solving DBP and assess its empirical performance. We show that the algorithm produces intuitively appealing basis vectors. On the other hand, the continuous decomposition methods often give better reconstruction accuracies and are also stronger in providing predictive features for classification. We discuss the reasons for this behavior.

In addition to DBP, we also define and consider a variation of it, namely, the *Discrete Basis Partitioning Problem* (DBPP). It adds a restriction to the type of feasible answers. While this restriction makes the answers somewhat less intuitive, and while the problem is still NP-hard, we show that, unlike DBP, DBPP can be approximated within a constant factor.

- *P. Miettinen and H. Mannila are with the Helsinki Institute for Information Technology, University of Helsinki and Helsinki University of Technology, PO Box 68 (Gustaf Hällströmin katu 2b), FI-00014 University of Helsinki, Finland. E-mail: {pamietti, heikki.mannila}@cs.helsinki.fi.*
- *T. Mielikäinen is with the Palo Alto Systems Research Center, Nokia, 955 Page Mill Road, Palo Alto, CA 94304. E-mail: taneli.mielikainen@nokia.com.*
- *A. Gionis is with Yahoo! Research, Barcelona, Ocata 1, 08003 Barcelona, Spain. E-mail: gionis@yahoo-inc.com.*
- *G. Das is with the Computer Science and Engineering Department, University of Texas at Arlington, 416 Yates Street, 302, Nedderman Hall, Arlington, TX 76019. E-mail: gdas@cse.uta.edu.*

Manuscript received 9 May 2007; revised 11 Dec. 2007; accepted 14 Feb. 2008; published online 5 Mar. 2008.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-2007-05-0205.

The rest of this paper is organized as follows: In Section 2, we formally define DBP and DBPP. In Section 3, we review related work, and in Section 4, we compare continuous and discrete matrix decomposition approaches. In Section 5, we discuss issues related to the computational complexity of DBP and DBPP. In Section 6, we present our greedy algorithm, and in Section 7, we report experimental results. Finally, Section 8 is a short conclusion.

2 Problem Definitions

Consider an $n \times m$ binary matrix $\mathbf{C}$. The rows of the matrix represent observations, and the columns represent the attributes of the data set. For instance, in a document corpus data set, rows are documents and columns are words, and $c_{ij} = 1$ denotes that the ith document contains the jth word.

A *basis vector*, intuitively, represents a set of correlated attributes. In the document corpus example, a basis vector corresponds to a set of words that constitutes a *topic*. The DBP formulation aims at discovering the topics that are present in the data set and also at discovering how each observation (document) in the data set can be expressed by a combination of those topics.

Let $\mathbf{S}$ and $\mathbf{B}$ be binary matrices of dimensions $n \times k$ and $k \times m$, respectively. Let $n \times m$ matrix $\mathbf{P} = \mathbf{S} \circ \mathbf{B}$ denote the Boolean product of $\mathbf{S}$ and $\mathbf{B}$, i.e., the matrix product with addition defined by $1 + 1 = 1$. The ith row of $\mathbf{P}$ is the logical OR of the rows of $\mathbf{B}$ for which the corresponding entry in the ith row of $\mathbf{S}$ is 1. Intuitively, $\mathbf{S}$ is the *usage matrix*, i.e., it contains information about which topics appear in each observation, and $\mathbf{B}$ is the *basis vector matrix*, i.e., it contains information about which attributes appear in each topic.

The goal in the DBP is to find k basis vectors $\mathbf{b} \in \{0,1\}^m$ such that the given data vectors $\mathbf{c} \in \{0,1\}^m$ can be represented (accurately) by using disjunctions of the basis vectors. The key aspect of the formulation is that both decomposition matrices, $\mathbf{S}$ and $\mathbf{B}$, are required to be binary and are thus more easily interpretable than arbitrary real matrices. Formally, DBP is defined as follows:

Problem 1 (the Discrete Basis Problem). *Given a binary $n \times m$ matrix $\mathbf{C}$ and a positive integer k, find an $n \times k$ binary matrix $\mathbf{S}$ and a $k \times m$ binary matrix $\mathbf{B}$ that minimize*

$$|\mathbf{C} - \mathbf{S} \circ \mathbf{B}| = \sum_{i=1}^{n} \sum_{j=1}^{m} \left| c_{ij} - (\mathbf{S} \circ \mathbf{B})_{ij} \right|. \quad (1)$$

DBPP differs from DBP by an extra constraint to the matrix $\mathbf{B}$: it is required to be a partition of the columns. An $n \times m$ binary matrix $\mathbf{B}$ is a *partition* (of the columns), if, for each column $\mathbf{b}_{\cdot j}$ of $\mathbf{B}$, there exists exactly one row $\mathbf{b}_i$ such that $b_{ij} = 1$. (Interpreting $\mathbf{B}$ as a collection of n sets over a universe of m elements should clarify the notion.) DBPP is given as follows:

Problem 2 (the Discrete Basis Partitioning Problem). *Given a binary $n \times m$ matrix $\mathbf{C}$ and a positive integer k, find an $n \times k$ binary matrix $\mathbf{S}$ and a $k \times m$ binary matrix $\mathbf{B}$ so that $\mathbf{B}$ is a partition, and $\mathbf{S}$ and $\mathbf{B}$ minimize*

$$\sum_{i=1}^{n} \sum_{j=1}^{m} \left| c_{ij} - (\mathbf{S} \circ \mathbf{B})_{ij} \right|. \quad (2)$$

The extra constraint introduced by DBPP potentially makes the decomposition less intuitive. For example, with corpus data it does not sound very intuitive to require that all words belong to some basis vector and that none of the words should belong to two or more basis vectors. Indeed, in this paper, DBPP has a theoretically oriented role. It will serve as a link between DBP and other better-known problems, as we will see in Sections 4 and 5.

3 Related Work

Probably the best-known method to decompose a matrix is the SVD [2]. It decomposes a matrix $\mathbf{A}$ into the form $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices and $\mathbf{\Sigma}$ is a diagonal matrix with positive entries—the singular values of $\mathbf{A}$. SVD gives the *optimal* rank-k approximation of the matrix $\mathbf{A}$ (simply by setting all but the k largest singular values to 0). Optimality of SVD means that the approximation produced by SVD is the best with respect to the squared reconstruction error and using normal matrix multiplication for real matrices. SVD has been widely used in data mining for matrix simplification and topic identification.

One problem with SVD is that the factor matrices can also contain negative values that are difficult to interpret (see also Section 4). In NMF, the factor matrices are required to have only nonnegative values. Early formulations of the NMF problem include those of Paatero and Tapper [6], where they call it "positive matrix factorization," and Cohen and Rothblum [7]. However, probably the most famous formulation is due to Lee and Seung [3]. Since their article, the problem has attained a lot of research and many algorithms are developed for it; for a recent survey, see [8].

In addition to SVD and NMF, many other matrix decomposition methods have been proposed, many of which are based on probabilistic models. Such methods include multinomial PCA [5], probabilistic Latent Semantic Indexing (PLSI) [9], and LDA [4]. All of these methods are related to a more general model, Gamma-Poisson model [10].

While the aforementioned methods are similar to DBP in many ways, they all assume non-Boolean data (i.e., real- or integer-valued data). Probabilistic methods assuming Boolean data include, for example, aspect Bernoulli models [11], topic models [12], and logistic PCA (LPCA) [13]. LPCA has similarities with DBP: Given an $n \times m$ data matrix $\mathbf{D}$ and an integer k, LPCA finds an $n \times k$ coefficient matrix $\mathbf{U}$ and a $k \times m$ basis vector matrix $\mathbf{V}$. However, unlike in DBP, these matrices are not Boolean. Instead, they are real-valued, and their product defines an element-wise probability distribution for Boolean $n \times m$ matrices: The probability that $d_{ij} = 1$ given the product $\mathbf{UV} = \mathbf{\Theta}$, $\Pr(d_{ij} = 1 | \Theta_{ij})$, is defined to be $\sigma(\Theta_{ij}) = (1 + \mathrm{e}^{-\Theta_{ij}})^{-1}$, i.e., the sigmoid (or logistic) function of Θ_{ij}.

Semidiscrete decomposition [14] lies between SVD and DBP. Like SVD, semidiscrete decomposition decomposes a given matrix into three matrices, $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$, but the values of $\mathbf{U}$ and $\mathbf{V}$ are restricted to -1, 0, and 1 only.

Semidiscrete decomposition was originally designed for image compression [14] but has since been used, e.g., in information retrieval [15].

CUR- and CX-type decompositions have recently attained some research interests (e.g., [16] and [17]). In a CUR decomposition, a matrix $\mathbf{D}$ is represented as a product of three matrices, $\mathbf{C}$, $\mathbf{U}$, and $\mathbf{R}$. Matrix $\mathbf{C}$ has c columns from $\mathbf{D}$, matrix $\mathbf{R}$ has r rows from $\mathbf{D}$, and matrix $\mathbf{U}$ is selected so that the product $\mathbf{CUR}$ is close to $\mathbf{D}$. In a CX decomposition, $\mathbf{D}$ is represented as a product $\mathbf{CX}$, where $\mathbf{C}$ again contains a subset of columns from $\mathbf{D}$, and $\mathbf{X}$ is an arbitrary coefficient matrix selected to minimize the representation error. For more information, see [18] and references therein.

Likewise, hierarchical descriptions of binary data have been studied: the Proximus framework constructs a hierarchical clustering of rows of a given binary matrix [19] and hierarchical tiles are probabilistic models hierarchically decomposing a binary matrix into almost monochromatic 0-1 submatrices [20].

Tiling transaction databases (i.e., binary matrices) is another line of related research [21]. A tiling covers a given binary matrix with a small number of submatrices full of 1s. The main difference to DBP is that no 0s can be covered in a feasible tiling. Methods have been developed for finding also large approximate tiles, for example fault-tolerant patterns [22] and conjunctive clusters [23], but obtaining an accurate description of the whole data set with a small number of approximate tiles has not been explicitly studied previously.

Boolean factorization, i.e., factoring Boolean functions [24], is an important part of logic synthesis. Rectangular coverings of Boolean matrices are one method used to obtain good factorizations. However, the weight functions used and the acceptance of noise are different to those of our work.

In coclustering (or biclustering), the goal is to cluster simultaneously both dimensions of a matrix [25]. A cocluster is thus a pair (R, C), R giving the indices of rows and C giving the indices of columns. Decomposing a Boolean matrix into two matrices can be seen as a coclustering of binary data where the clusters can overlap. The idea of coclustering was originally proposed by Hartigan in 1972 [25], but it has gained a lot of attention recently, and many new methods have been proposed; see, for example, [26], [27], and [28].

After the publication of a preliminary version of this paper [1], Vaidya et al. [29] have shown how the DBP lends itself to role-based access control framework. Vaidya et al. show that the DBP as we present it here is exactly the Minimal Noise Role Mining Problem; the latter problem is defined by Vaidya et al. and claimed to have pragmatic implications [29]. In addition, in a recent paper related to [29], Lu et al. [30] formulate DBP as an integer programming problem with exponential number of variables and constraints.

4 Continuous and Discrete Decompositions

In this section, we discuss the properties of continuous and discrete approaches to matrix decomposition, and in particular, the properties of SVD as compared to those of DBP. Most of this section is about matrix ranks. Different decompositions induce different ranks, and these induced ranks are an important tool on studying the relationships between decompositions.

4.1 Matrix Decompositions and Matrix Ranks

Before going to the ranks, let us study a simple example demonstrating some properties of SVD and DBP. In SVD, the resulting matrices, $\mathbf{U}$ and $\mathbf{V}$, have real-valued and even negative entries, so they do not necessarily have an intuitive interpretation. For obtaining intuition from a concrete example, consider the toy data set representing student enrollments in university courses, as in the example mentioned in the Introduction. In particular, let $\mathbf{C}$ be a 3×3 Boolean matrix, where the rows represent the students X, Y, and Z, and the columns denote the courses Operating systems, Programming languages, and Compilers:

$$\mathbf{C} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix}.$$

The rank-2 SVD of $\mathbf{C}$ is

$$\mathbf{U\Sigma V}^T = \begin{pmatrix} 0.50 & 0.71 \\ 0.71 & 0 \\ 0.50 & -0.71 \end{pmatrix} \begin{pmatrix} 2.41 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0.50 & 0.71 \\ 0.71 & 0 \\ 0.50 & -0.71 \end{pmatrix}^T.$$

The basis vectors in $\mathbf{V}$ are not easy to interpret. Matrix $\mathbf{C}$ has rank 3, and the approximation to $\mathbf{C}$ produced by SVD with rank-2 decomposition is

$$\mathbf{U\Sigma V}^T = \begin{pmatrix} 1.10 & 0.85 & 0.10 \\ 0.85 & 1.21 & 0.85 \\ 0.10 & 0.85 & 1.10 \end{pmatrix}.$$

By the optimality of SVD, this is the best that can be achieved by looking at real matrices of rank 2 and squared error.

On the other hand, DBP produces the representation

$$\mathbf{C} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{pmatrix} \circ \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix},$$

which has no error and is easy to understand. In our example's framework, the columns of the first factor matrix assign students to different specialization areas and the rows of the second factor matrix define the courses required in each specialization area.

As noted above, SVD produces optimal $\mathrm{rank}\text{-}k$ representations of matrices with respect to the Frobenius norm (sum of squares of elements). It is also relatively fast to compute, requiring time $O(nm \min\{n, m\})$ [2].

Optimality for arbitrary matrices is not the whole story, however. For binary matrices, one can study different types of ranks. The *real rank* $r_R(\mathbf{C})$ of a binary matrix $\mathbf{C}$ is simply the smallest value of k such that $\mathbf{C} = \mathbf{SB}$ with an $n \times k$ matrix $\mathbf{S}$, a $k \times m$ matrix $\mathbf{B}$, and using normal matrix multiplication. The *nonnegative rank* $r_{R^+}(\mathbf{C})$ of $\mathbf{C}$ is similar to the real rank, but the factor matrices are restricted to have only nonnegative values. The *nonnegative integer rank* $r_N(\mathbf{C})$ of $\mathbf{C}$ further restricts the factor matrices $\mathbf{S}$ and $\mathbf{B}$: In

nonnegative integer rank, matrices $\mathbf{S}$ and $\mathbf{B}$ can only contain nonnegative integers. Finally, the *Boolean rank* $r_B(\mathbf{C})$ of $\mathbf{C}$ is the smallest k such that $\mathbf{C} = \mathbf{S} \circ \mathbf{B}$, where $\mathbf{S}$ is an $n \times k$ binary matrix, $\mathbf{B}$ is a $k \times m$ binary matrix, and the matrix multiplication is Boolean.

The Boolean rank is thus the smallest k such that we can solve DBP with zero error, the real rank is the smallest k such that SVD will give zero error, and the nonnegative rank is the smallest k such that NMF can give zero error. For DBPP, it follows from the requirement of the matrix $\mathbf{B}$ being a partition that the normal arithmetic could be used in the matrix multiplication: all of the summations involved in the matrix multiplication can have at most one nonzero term. Thus, the nonnegative integer rank is a lower bound for k such that we can solve DBPP with zero error. It is, however, only a lower bound: The requirement that $\mathbf{B}$ is a partition is a sufficient, yet not necessary, condition to make sure that the product $\mathbf{SB}$ is a binary matrix.

The concepts of real and Boolean ranks discuss the exact representation of the matrix $\mathbf{C}$, and we are more interested in the approximate representation. One could define the ε-ranks $r_R^\varepsilon(\mathbf{C})$, $r_{R^+}^\varepsilon(\mathbf{C})$, $r_N^\varepsilon(\mathbf{C})$, and $r_B^\varepsilon(\mathbf{C})$ to be otherwise as their non-ε versions, but instead of an exact decomposition, they ask for a decomposition that does not cause more than ε error in the reconstruction. For example, the $r_B^\varepsilon(\mathbf{C})$ of an $n \times m$ binary matrix $\mathbf{C}$ is the smallest k such that $|\mathbf{C} - \mathbf{S} \circ \mathbf{B}| \leq \varepsilon$ with $\mathbf{S} \in \{0,1\}^{n \times k}$ and $\mathbf{B} \in \{0,1\}^{k \times m}$.

4.2 Relations between Ranks

There has been some study with respect to the relations of these ranks (cf. [31], [32]). In particular, the following inequalities hold for the nonnegative rank of a binary matrix $\mathbf{C}$ [32]:

$$r_R(\mathbf{C}) \leq r_{R^+}(\mathbf{C}) \tag{3}$$

and

$$r_B(\mathbf{C}) \leq r_{R^+}(\mathbf{C}). \tag{4}$$

Similar inequalities hold also for the nonnegative integer rank of a binary matrix $\mathbf{C}$ [32]:

$$r_R(\mathbf{C}) \leq r_N(\mathbf{C}) \tag{5}$$

and

$$r_B(\mathbf{C}) \leq r_N(\mathbf{C}). \tag{6}$$

Inequality (5) follows because both ranks use the same arithmetic, and (6) follows because the factor matrices are binary in both cases.

Between the real and Boolean ranks, there are no clear relations. It can be shown that there are binary matrices $\mathbf{C}$ for which $r_R(\mathbf{C}) < r_B(\mathbf{C})$ and vice versa [31]. The complement of the identity matrix of size $n \times n$ is an example where $r_B(\mathbf{C}) = O(\log n)$, but $r_R(\mathbf{C}) = n$ [31]. This shows that while SVD can use the space of real ranks, DBP can take advantage of the properties of Boolean operations to achieve much smaller rank than SVD. Empirical experiments on generated data support this conclusion. Thus, it is not a priori obvious that SVD will produce more concise representations than the Boolean methods.

Computing the real rank is easy (excluding the precision issues) and can be done, e.g., using SVD: The real rank of a matrix is the number of its nonzero singular values. Computing the Boolean rank, however, is NP-complete: identifying a binary matrix as an adjacency matrix of some bipartite graph G, the Boolean rank of that matrix is exactly the number of complete bipartite subgraphs needed to cover all edges of G [31]. This problem, covering by complete bipartite subgraphs, is well known to be NP-complete [33, problem GT18].

Approximating Boolean rank is also hard. In [34], it is shown to be as hard to approximate as the problem of partitioning a graph into cliques [33, problem GT15] (equivalently, as hard to approximate as the minimum chromatic number). Yet, there exist some upper and lower bounds for the Boolean rank, and the relation between the Boolean and real ranks is known in some special cases; see [31] and references therein. Finally, the problem "Given $\mathbf{C}$ and k, is $r_B(\mathbf{C}) \leq k$?" is *fixed-parameter tractable*, i.e., it can be solved in time polynomial to the size of $\mathbf{C}$, assuming that k is fixed [35], [36].

In the case of ε-ranks, inequality (4) does not hold, but inequality (3) does hold. For nonnegative integer rank, the upper bound property of course carries over in both cases:

$$r_R^\varepsilon(\mathbf{C}) \leq r_N^\varepsilon(\mathbf{C}) \tag{7}$$

and

$$r_B^\varepsilon(\mathbf{C}) \leq r_N^\varepsilon(\mathbf{C}). \tag{8}$$

In other words, knowing that one can solve DBPP with parameter k and error ε, one knows that the same error is attainable in DBP with parameter $k' \leq k$, or with the same parameter, the error $\varepsilon' \leq \varepsilon$ can be achieved. Similar results also hold for the real-valued decompositions.

Otherwise, even less seems to be known about ε-ranks than about exact ranks. One goal of this paper is to investigate empirically and theoretically whether the Boolean decompositions are feasible alternatives of the continuous methods.

5 Computational Complexity

In this section, we study the computational complexity of DBP and DBPP. We show that both of the problems are NP-hard to solve exactly and prove that DBP is NP-hard to approximate within any factor, while DBPP can be approximated within a constant factor. Especially, we show how DBPP is related to the well-known Metric k-Median Problem; our results concerning DBPP then follow from this relation.

5.1 The Discrete Basis Problem

The DBP is an optimization problem: Find the matrix decomposition into k basis vectors that minimizes the representation error according to the definition of Problem 1. To put the problem in the perspective of complexity theory, we formulate the decision version of the problem. This is defined as in Problem 1, but additionally, we are given a target cost t and the task is to decide whether there is a decomposition of the input binary matrix $\mathbf{C}$ into binary matrices $\mathbf{S}$ and $\mathbf{B}$ that yields an error at most equal to t.

The NP-completeness of the decision version of DBP could be proved by a reduction from the problem of computing the Boolean rank. However, we will give a reduction from a different NP-complete problem, the Set Basis Problem (SBP) [33, problem SP7].

Problem 3 (SBP). *Given a collection $\mathcal{C}$ of subsets of a finite universe U and a positive integer k, decide whether or not there is a collection $\mathcal{B} \subseteq 2^U$ of at most k sets ($|\mathcal{B}| \leq k$) such that for every set $C \in \mathcal{C}$ there is a subcollection $\mathcal{B}_C \subseteq \mathcal{B}$ with $\bigcup_{B \in \mathcal{B}_C} B = C$.*

We will show that SBP is a special case of DBP, and that the decision version of DBP is in NP.

Theorem 1. *The decision version of DBP is NP-complete.*

Proof. For any instance of SBP, there is an equivalent instance of DBP with $t = 0$, even when only the matrix $\mathbf{B}$ is requested. Since the exact decomposition is requested, if we are given matrix $\mathbf{B}$, we can construct $\mathbf{S}$ row by row in polynomial time simply by selecting greedily all those rows of $\mathbf{B}$ that do not cover any 0s in input data. If $\mathbf{B}$ is indeed a part of an exact decomposition, then this greedy procedure will result in correct $\mathbf{S}$. Hence, DBP is NP-hard.

Finally, it is immediate that the decision version of DBP is in NP, because the sizes of $\mathbf{B}$ and $\mathbf{S}$ are polynomial in the size of $\mathbf{C}$. □

The reduction from SBP to DBP with $t = 0$ implies also the following inapproximability result:

Theorem 2. *DBP cannot be approximated within any factor in polynomial time, unless* $\mathrm{P} = \mathrm{NP}$.

Note that solving SBP is equivalent to computing the Boolean rank: The result of SBP with parameter k is "yes" if and only if the Boolean rank of the matrix is at most k. Likewise, finding the set basis is (polynomial-time) equivalent to finding the exact Boolean decomposition: set basis is exactly the matrix $\mathbf{B}$, and given $\mathbf{B}$, the matrix $\mathbf{S}$ can be solved in polynomial time if we want an exact decomposition.

The problem of solving the whole approximate decomposition of the matrix for given basis vectors, i.e., finding the matrix $\mathbf{S}$ for given $\mathbf{B}$ and $\mathbf{C}$, can be solved by a straightforward algorithm in time $O(2^k mn)$, where k is the number of basis vectors (i.e., the number of rows in $\mathbf{B}$): each of the n rows in $\mathbf{C}$ can be decomposed independently and there are only 2^k different ways to choose a subset of basis vectors. Thus, the problem of finding the optimal decomposition, given that a basis vector matrix is known, is also fixed-parameter tractable (see [35] and [36]).

5.2 The Discrete Basis Partitioning Problem

A main source of difficulties in DBP seems to be in the overlapping basis vectors. Hence, it is natural to ask whether the problem becomes easier if the basis vectors do not overlap. Such a variant of DBP is the DBPP. To see that DBPP can indeed be easier than DBP, consider the case where no error is allowed. Then, DBP is NP-hard, but DBPP can be solved in time linear in the number of ones in the input matrix since the basis vectors are exactly the maximal sets of identical columns.

First, we express DBPP as a clustering problem: the Binary k-Median Problem (BKMP). We will show that DBPP and BKMP are equivalent problems, in the sense that all results regarding the computational complexity and approximation of one of these problems carry over to the other. We can then consider only the complexity and approximation of BKMP, obtaining simultaneously the same answers for DBPP.

Problem 4 (BKMP). *Given a set C of Boolean m-dimensional vectors with $|C| = n$ and a positive integer k, find a set $M = \{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$ of Boolean m-dimensional vectors (the medians), and a partition $\mathcal{D} = \{D_1, \ldots, D_k\}$ of C such that M and $\mathcal{D}$ minimize*

$$\sum_{j=1}^{k} \sum_{\mathbf{c} \in D_j} \|\mu_j - \mathbf{c}\|_1, \tag{9}$$

where $\|\cdot\|_1$ is the Hamming norm.

Note that in the above definition, the vectors $\boldsymbol{\mu}_j$ are allowed to be arbitrary Boolean vectors, i.e., M does not have to be a subset of C. If this is not the case, i.e., $M \subseteq C$ is required, the problem is known as the (Binary) Metric k-Median Problem. The Metric k-Median Problem will be our starting point for studying the complexity of BKMP. However, before that, we prove the aforementioned connection between DBPP and BKMP.

We show that DBPP and BKMP are equivalent, that is, given an instance of one of the two problems we can transform it to an instance of the other problem having a solution of the same cost.

Theorem 3. *DBPP and BKMP are equivalent problems.*

Proof. We show that given an instance of each of the problems we can transform it to an instance of the other problem having a solution of the same cost.

First, we define the mapping for the inputs. Given an input matrix $\mathbf{C}$ and an integer k, the mapping simply transposes $\mathbf{C}$ and keeps k intact. It can be used to transform an instance of DBPP to an instance of BKMP and vice versa.

Second, we consider how to map the solutions from one problem to the other. Let $\mathbf{S}$ and $\mathbf{B}$ be two matrices defining a solution for the given instance of DBPP. Identify set M as a $k \times m$ matrix $\mathbf{M}$. We define $\mathbf{M}$ to be the transpose of $\mathbf{S}$, and the sets of the partition $\mathcal{D}$ to be such that the vector $\mathbf{c}'_j$ belongs into the set $D_i \in \mathcal{D}$ if and only if $(\mathbf{B})_{ij} = 1$.

Hence, the idea is that the instances and feasible solutions of BKMP are the transposes of those of DBPP. The costs of solutions are equal since the cost of BKMP is just

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \left| (\mathbf{C}^T)_{ij} - (\mathbf{B}^T \circ \mathbf{S}^T)_{ij} \right|, \tag{10}$$

due to the definitions of the mappings. □

We approach the computational complexity of DBPP by studying the computational complexity of BKMP. Again, we need to consider the decision version of BKMP: we are given

an instance of BKMP and a target cost t, and the question is, is there a solution to the BKMP instance with cost at most t?

Lemma 1. *The decision version of the BKMP is NP-complete.*

Proof. First, notice that BKMP belongs in NP.

As a notational convention, we use $[N]$ to denote the set $\{0, 1, \ldots, N\}$ throughout this proof. To prove the NP-hardness, we show a reduction to BKMP from the following problem:

Problem 5. *Let k and t be positive integers and let A be a set of n 2D points so that if (a_1, a_2), $(a'_1, a'_2) \in A$, then*

$$\{|a_i - a'_i| : i = 1, 2\} \subseteq [N], \tag{11}$$

where N is polynomially bounded by n. Is there a set $B = \{\mathbf{b}_1, \ldots, \mathbf{b}_k\}$ of 2D points and a partition $\mathcal{E} = \{E_1, \ldots, E_k\}$ of A such that

$$\sum_{j=1}^{k} \sum_{\mathbf{a} \in E_j} \|\mathbf{b}_j - \mathbf{a}\|_1 \leq t? \tag{12}$$

Problem 5 was used by Megiddo and Supowit [37] to prove the NP-hardness of the 2D k-median problem. Even though Megiddo and Supowit do not explicitly state Problem 5, they prove that it is NP-complete.

We now show how to reduce an instance (A, k, t) of Problem 5 to an instance (C, k, t) of the decision version of BKMP so that the answer to (C, k, t) is "yes" if and only if the answer to (A, k, t) is "yes."

Consider an instance of Problem 5, i.e., a triplet (A, k, t). Without loss of generality, we may assume that the points in A are in the nonnegative quadrant of the real plane. Furthermore, due to restriction (11), we may assume that $A \subseteq [N] \times [N]$.

We note that the sum of the Hamming distances is minimized by the coordinate-wise median of the points in the sum. That is, for any solution $(B, \mathcal{E})$ minimizing (12), we may assume that $B \subseteq [N] \times [N]$.

For the reduction, we need to embed the points of A into a Boolean space. To obtain such an embedding, notice that

$$\{\|\mathbf{a} - \mathbf{a}'\|_1 : \mathbf{a}, \mathbf{a}' \in A\} \subseteq [2N].$$

We can use the standard technique of writing the coordinates of points of A in unary in order to embed A into an $8N$-dimensional Boolean space. It is straightforward to see that this embedding does not change the Hamming distance between the points. Furthermore, since N is polynomially bounded in n, the reduction is polynomial.

Finally, consider a solution $(M, \mathcal{D})$ of instance (C, k) so that (9) evaluates to at most t. Because points in C represent two integers written in unary, and because we can assume that the points in M are coordinate-wise medians of subsets of points in C, we see that also points in M represent two integers written in unary. Thus, using the reverse of the above reduction, we can have a solution $(B, \mathcal{E})$ of the original instance (A, k) so that (12) evaluates to at most t. □

The following two corollaries are immediate implications of Theorem 3 and Lemma 1.

Corollary 1. *DBPP is NP-hard.*

Corollary 2. *If we can approximate BKMP within a factor r, then we can approximate DBPP within the same approximation factor and vice versa.*

The BKMP can be approximated within a factor of at most 10. To see this, recall that in the (Binary) Metric k-Median Problem, set M is required to be a subset of set C. An answer to the Binary Metric k-Median Problem is clearly a valid answer to BKMP, too, and—due to the triangle inequality—the error of an optimal answer to it is at most twice as large as the error of the optimal answer to BKMP.

Thus, an approximation algorithm for the Metric k-Median Problem with an approximation factor r is an approximation algorithm for BKMP with an approximation factor $2r$. The claim now follows, because for example Arya et al. [38] have proposed an approximation algorithm for the Metric k-Median Problem with an approximation ratio of at most 5.

6 The Algorithms

In this section, we give an algorithm for DBP. We will not give an algorithm for DBPP, since, as Corollary 2 stated, we can use any of the existing algorithms for the Metric k-Median Problem to solve it. Section 6.1 describes the basic version of the algorithm, and Section 6.2 introduces some simple improvements to it.

6.1 The Basic Algorithm

We will now give a simple greedy algorithm for DBP. The basic idea of the algorithm is to exploit the correlations between the columns: First, the associations between every two columns are computed. Second, the associations are used to form candidate basis vectors. Third, a small set of candidate basis vectors is selected in a greedy way to form the basis. The algorithm is described in pseudocode in Algorithm 1 below.

In the rest of the section, we denote a row vector of a matrix $\mathbf{M}$ by $\mathbf{m}_i$, a column vector by $\mathbf{m}_{\cdot i}$, and a matrix entry by m_{ij}. Furthermore, if $\mathbf{M}$ has n rows and $I \subseteq \{1, \ldots, n\}$, then $\mathbf{M}_I$ is the submatrix of $\mathbf{M}$ that contains the rows $\mathbf{m}_i$, $i \in I$. The indicator function $\mathbf{1}(P)$ takes a value of 1 if proposition P is true and 0 otherwise.

The confidence of an association between the ith and jth columns of matrix $\mathbf{C}$ is defined as in association rule mining [39], i.e., $c(i \Rightarrow j, \mathbf{C}) = \langle \mathbf{c}_{\cdot i}, \mathbf{c}_{\cdot j} \rangle / \langle \mathbf{c}_{\cdot i}, \mathbf{c}_{\cdot i} \rangle$, where $\langle \cdot, \cdot \rangle$ is the vector inner product operation. An association between columns i and j is τ-strong if $c(i \Rightarrow j, \mathbf{C}) \geq \tau$.

Algorithm 1 An algorithm for the DBP (ASSO).

Input: A matrix $\mathbf{C} \in \{0, 1\}^{n \times m}$ for data, a positive integer k, a threshold value $\tau \in]0, 1]$, and real-valued weights w^+ and w^-.

Output: Matrices $\mathbf{B} \in \{0, 1\}^{k \times m}$ and $\mathbf{S} \in \{0, 1\}^{n \times k}$.

```
1: function ASSO (C, k, τ, w+, w−)
2:     for i = 1, ..., m do  ▷ Construct matrix A row by row.
3:         a_i ← (1(c(i ⇒ j, C) ≥ τ))_{j=1}^m
4:     B ← [ ], S ← [ ]  ▷ B and S are empty matrices.
```

5: **for** $l = 1, \ldots, k$ **do** ▷ Select the k basis vectors from $\mathbf{A}$.
6: $(\mathbf{a}_i, \mathbf{s}) \leftarrow \arg\max_{\mathbf{a}_i, \mathbf{s} \in \{0,1\}^{n \times 1}} cover\left(\begin{bmatrix}\mathbf{B}\\ \mathbf{a}_i\end{bmatrix}, [\mathbf{S}\ \ \mathbf{s}], \mathbf{C}, w^+, w^-\right)$
7: $\mathbf{B} \leftarrow \begin{bmatrix}\mathbf{B}\\ \mathbf{a}_i\end{bmatrix}$, $\mathbf{S} \leftarrow [\mathbf{S}\ \ \mathbf{s}]$
8: **return** $\mathbf{B}$ and $\mathbf{S}$

We construct an *association matrix* $\mathbf{A}$ where row $\mathbf{a}_i$ consists of 1's in columns j such that $c(i \Rightarrow j, \mathbf{C}) \geq \tau$. Each row of $\mathbf{A}$ is considered as a candidate for being a basis vector. The threshold τ controls the level of confidence required to include an attribute to the basis vector candidate, and it is assumed that τ is a parameter of the algorithm.

The DBP objective function (1) penalizes equally for both types of errors: for 0 becoming 1 in the approximation, and for 1 becoming 0. We have found that in practice, the results of our algorithm can be improved if we distinguish between these two types of error. Thus, we introduce weights w^+ and w^- that are used to reward for covering 1s and penalize for covering 0s, respectively. Clearly, without loss of generality, we can assume that $w^- = 1$.

The basis vectors are selected from the matrix $\mathbf{A}$ and the columns of the usage matrix $\mathbf{S}$ are fixed in a greedy manner as follows: Initially, $\mathbf{S}$ and $\mathbf{B}$ are empty matrices. The basis $\mathbf{B}$ is updated in the iteration l by adding the lth row $\mathbf{b}_l$, and matrix $\mathbf{S}$ is updated by adding the lth column $\mathbf{s}_{\cdot l}$. The row $\mathbf{b}_l$ is a row $\mathbf{a}_i$ from $\mathbf{A}$ and the column $\mathbf{s}_{\cdot l}$ is an arbitrary n-dimensional binary column vector. The selection of $\mathbf{b}_l$ and $\mathbf{s}_{\cdot l}$ is done so to maximize $cover(\mathbf{B}, \mathbf{S}, \mathbf{C}, w^+, w^-)$, which is defined to be equal to

$$w^+ \left|\left\{(i,j) : c_{ij} = 1, (\mathbf{S} \circ \mathbf{B})_{ij} = 1\right\}\right| - w^- \left|\left\{(i,j) : c_{ij} = 0, (\mathbf{S} \circ \mathbf{B})_{ij} = 1\right\}\right|.$$

The value of function *cover* can be considered as the "profit" of describing $\mathbf{C}$ using matrices $\mathbf{B}$ and $\mathbf{S}$.

The association matrix can be constructed in time $O(nm^2)$, and a single discrete basis vector can be obtained in time $O(nm^2)$. Thus, Algorithm 1 has time complexity $O(knm^2)$.

The algorithm has two parameters that control the quality of results: the threshold τ and weight w^+ (again assuming that $w^- = 1$). The straightforward way to set the parameters is to try several different possibilities and take the one that minimizes the reconstruction error. Alternatively, the weight w^+ can be used to express different valuations for covering 1s and 0s.

Why association confidences, i.e., why do we consider the rows of the matrix $\mathbf{A}$ as candidate basis vectors? To see the intuition, consider a Boolean matrix $\mathbf{C}$ that has an exact decomposition $\mathbf{S} \circ \mathbf{B}$. Let $\mathbf{b}_p$ be the pth row of $\mathbf{B}$ with $b_{pi} = 1$ and $b_{pj} = 1$ for some i and j. Let q be such that $s_{qp} = 1$. Then, we know that $c_{qi} = c_{qj} = 1$ for all such q. If no other row of $\mathbf{B}$ has 1 in positions i and j, then $c(i \Rightarrow j, \mathbf{C}) = c(j \Rightarrow i, \mathbf{C}) = 1$. Then, consider a more complex case when there is another row of $\mathbf{B}$, say r, that has $b_{rj} = 1$, but still $b_{ri} = 0$. If there is also a row q of $\mathbf{S}$ with $s_{qp} = 0$ and $s_{qr} = 1$, then the confidence of the rule $j \Rightarrow i$ is no longer 1. However, it still holds that $c(i \Rightarrow j, \mathbf{C}) = 1$. Thus, in a case with no noise the row $\mathbf{b}_p$ of $\mathbf{B}$ is a row of the matrix $\mathbf{A}$, given that there is i so that $b_{pi} = 1$ and $b_{qi} = 0$ for all $q \neq p$.

Unfortunately, that is the best we can do: If for all i so that $b_{pi} = 1$ there is q so that also $b_{qi} = 1$, then we cannot find row $\mathbf{b}_p$ from $\mathbf{A}$. A concrete example of this is given by the complement of the $n \times n$ identity matrix, $\bar{\mathbf{I}}_n$. The Boolean rank of $\bar{\mathbf{I}}_n$ is $O(\log n)$ and its real rank is n [31] (see also Section 4). However, with $k < n$, no value of τ will give a perfect decomposition with that input, as the association confidence matrix $\mathbf{A}'$ used by Algorithm 1 is

$$\mathbf{A}' = \begin{pmatrix} 1 & \frac{n-2}{n-1} & \frac{n-2}{n-1} & \\ \frac{n-2}{n-1} & 1 & \frac{n-2}{n-1} & \cdots \\ \frac{n-2}{n-1} & \frac{n-2}{n-1} & 1 & \\ & \vdots & & \ddots \end{pmatrix}.$$

Notice that Algorithm 1 produces matrix $\mathbf{A}$ from $\mathbf{A}'$ by letting $a_{ij} = \mathbf{1}(a'_{ij} \geq \tau)$, i.e., it will produce either a matrix full of 1's or an identity matrix, depending on the value of τ.

6.2 Improving the Algorithm

Though Algorithm 1 is very simple, it gives results that are comparable to those of more complex methods, as we will see in Section 7. In this section, we discuss four ideas that further improve the solutions obtained by Algorithm 1. The first two of the ideas rely on preprocessing the data before using Algorithm 1, while the last two ideas modify the algorithm itself. Henceforth, Algorithm 1 is referred to as the ASSO algorithm.

Our first improvement relies on the observation that for the decomposition $\mathbf{D} = \mathbf{S} \circ \mathbf{B}$ we have $\mathbf{D}^T = \mathbf{B}^T \circ \mathbf{S}^T$. In other words, in a discrete basis decomposition of the transpose $\mathbf{D}^T$ of the data matrix, the *usage matrix* of the decomposition gives the basis vectors for the matrix $\mathbf{D}$. Since the ASSO algorithm is heuristic, it is possible that when applied on the transpose matrix $\mathbf{D}^T$ it gives a better solution. Running the ASSO algorithm on $\mathbf{D}^T$ is denoted as ASSO + trans, and it is presented in Algorithm 2. In practice, one should of course run both ASSO and ASSO + trans and select the result that yields a smaller reconstruction error.

Algorithm 2 An algorithm for transposed DBP (ASSO + trans).
Input: A matrix $\mathbf{C} \in \{0,1\}^{n \times m}$ for data, a positive integer k, a threshold value $\tau \in]0, 1]$, and real-valued weights w^+ and w^-.
Output: Matrices $\mathbf{B} \in \{0,1\}^{k \times m}$ and $\mathbf{S} \in \{0,1\}^{n \times k}$.
1: **function** ASSOTRANS $(\mathbf{C}, k, \tau, w^+, w^-)$
2: $\mathbf{C} \leftarrow \mathbf{C}^T$
3: **return** ASSO $(\mathbf{C}, k, \tau, w^+, w^-)$

Our second modification of the basic algorithm stems from the observation that for data that are easily clustered, the basis vectors should be different among the different clusters. To take advantage of this fact, we propose to first cluster the data and then solve DBP within each cluster. The

basis vectors can be searched for each cluster independently. However, there can be some structure shared between the clusters. For example, in the course enrollment data, one expects to find also some study modules shared among the computer science, mathematics, and physics students.

In order to take the intra- and intercluster structure into account, we propose the following approach. We compose the set of candidate basis vectors from association rules within each cluster. Note that this is different to computing the association rules over the whole data, as the association strengths can vary considerably. We then select the final basis vectors from these candidates one by one. To select a basis vector, we first select the cluster that has the most uncovered 1s. The basis vector selected is the vector that maximizes the function *cover* within the selected cluster. Finally, we use that basis vector to cover the rest of the data.

In general, we do not fix the clustering method used in line 2 of Algorithm 3. It should be selected by the user according to her needs. Algorithm 3 finds the basis vector candidates for the clusters in the same way as in ASSO.

Algorithm 3 A DBP algorithm using clustering (ASSO + clust).

Input: A matrix $\mathbf{C} \in \{0,1\}^{n \times m}$ for data, a positive integer p for the number of clusters, a positive integer $k \geq p$, a threshold value $\tau \in]0,1]$, and real-valued weights w^+ and w^-.

Output: Matrices $\mathbf{B} \in \{0,1\}^{k \times m}$ and $\mathbf{S} \in \{0,1\}^{n \times k}$, and row indices for each cluster.

1: **function** ASSOCLUST $(\mathbf{C}, p, k, \tau, w^+, w^-)$
2: $(I_1, \ldots, I_p) \leftarrow$ CLUSTER$(\mathbf{C}, p)$ ▷ I_h's contain indices for $\mathbf{C}$'s rows.
3: $\mathbf{A} \leftarrow [\,], \mathbf{B} \leftarrow [\,], \mathbf{S} \leftarrow [\,]$
4: **for** $h = 1, \ldots, p$ **do**
5: **for** $i = 1, \ldots, m$ **do** ▷ Construct the basis vector candidate matrix $\mathbf{A}^{(h)}$ for each cluster h.
6: $\mathbf{a}_i^{(h)} \leftarrow (\mathbf{1}(c(i \Rightarrow j, \mathbf{C}_{I_h}) \geq \tau))_{j=1}^m$
7: $\mathbf{A} \leftarrow \begin{bmatrix} \mathbf{A} \\ \mathbf{A}^{(h)} \end{bmatrix}$
8: **for** $l = 1, \ldots, k$ **do**
9: $h \leftarrow \arg\max_{h=1,\ldots,p} |\{(i,j) : (\mathbf{C}_{I_h})_{ij} = 1, (\mathbf{S}_{I_h} \circ \mathbf{B})_{ij} = 0\}|$
10: $(\mathbf{a}_i, \mathbf{s}) \leftarrow \arg\max_{\mathbf{a}_i, \mathbf{s}} cover\left(\begin{bmatrix} \mathbf{B} \\ \mathbf{a}_i \end{bmatrix} [\mathbf{S}_{I_h} \;\; \mathbf{s}], \mathbf{C}_{I_h}, w^+, w^-\right)$
11: $\mathbf{B} \leftarrow \begin{bmatrix} \mathbf{B} \\ \mathbf{a}_i \end{bmatrix}$, $\mathbf{S}_{I_h} \leftarrow [\mathbf{S}_{I_h} \;\; \mathbf{s}]$
12: $J \leftarrow \{1, \ldots, n\} \setminus I_h$
13: $\mathbf{s} \leftarrow \arg\max_{\mathbf{s}} cover(\mathbf{B}, [\mathbf{S}_J \;\; \mathbf{s}], \mathbf{C}_J, w^+, w^-)$
14: $\mathbf{S}_J \leftarrow [\mathbf{S}_J \;\; \mathbf{s}]$
15: **return** $\mathbf{B}$, $\mathbf{S}$, and $(I_1, \ldots, I_p)$

The aforementioned ideas, ASSO + trans and ASSO + clust, are based on preprocessing the data. We now discuss two ideas that modify the way that the algorithm constructs the matrix $\mathbf{S}$. Our first method, ASSO + opt, solves this problem in an optimal manner. Recall from Section 5.1 that given $\mathbf{C}$ and $\mathbf{B}$, finding $\mathbf{S}$ such that $|\mathbf{C} - \mathbf{S} \circ \mathbf{B}|$ is minimized can be done in time exponential only with respect to k, the number of rows in $\mathbf{B}$. The algorithm, presented in Algorithm 4, performs such an exhaustive search: for each row in a data matrix, it tries all possible combinations of basis vectors, yielding a time complexity of $O(2^k nm)$.

Algorithm 4 A DBP algorithm using exhaustive search (ASSO + opt).

Input: A matrix $\mathbf{C} \in \{0,1\}^{n \times m}$ for data, a positive integer k, a threshold value $\tau \in]0,1]$, and real-valued weights w^+ and w^-.

Output: Matrices $\mathbf{B} \in \{0,1\}^{k \times m}$ and $\mathbf{S} \in \{0,1\}^{n \times k}$.

1: **function** ASSOOPT $(\mathbf{C}, k, \tau, w^+, w^-)$
2: $\mathbf{B} \leftarrow$ ASSO$(\mathbf{C}, k, \tau, w^+, w^-)$
3: **for** $i = 1, \ldots, n$ **do**
4: $\mathbf{s}_i \leftarrow \arg\min_{\mathbf{s} \in \{0,1\}^k} \sum_{j=1}^m |c_{ij} - (\mathbf{s} \circ \mathbf{B})_j|$
5: **return** $\mathbf{B}$ and $\mathbf{S}$

Notice that there is a considerable difference in the computational complexity between finding a single column of matrix $\mathbf{S}$ and finding a single row of $\mathbf{S}$. Assume first that we want to find a single column of $\mathbf{S}$, keeping other columns fixed. Each row of data matrix $\mathbf{C}$ can be considered independently and thus finding a single column of $\mathbf{S}$ maximizing *cover* is relatively easy. This is done, e.g., in the ASSO algorithm. However, no efficient algorithm is known to find a single row of $\mathbf{S}$ maximizing *cover* (equivalently, minimizing reconstruction error), and so ASSO + opt has to try all possibilities.

The final method, ASSO + iter, uses an iterative method to improve $\mathbf{S}$. It starts by calling the standard ASSO algorithm for a matrix $\mathbf{B}$ and an initial matrix $\mathbf{S}$. It then changes the columns of $\mathbf{S}$ so that the overall error decreases and continues until the error does not decrease anymore, i.e., until the algorithm has converged to a local optimum. Pseudocode is provided in Algorithm 5. If there are many possibilities for changing a column in $\mathbf{S}$ that yield the same error, the algorithm selects the one with the least number of 1s, thus achieving a sparser matrix. Another detail is that when selecting new columns for $\mathbf{S}$ in line 5, Algorithm 5 sets both weights w^+ and w^- to 1. In this case, the weights are used only to find better basis vectors, but the error is still considered to be symmetric, i.e., not weighted. If a weighted error function is used, then 1s should be replaced by w^+ and w^- when computing *cover* in line 5 of Algorithm 5.

Algorithm 5 A DBP algorithm using iterative search (ASSO + iter).

Input: A matrix $\mathbf{C} \in \{0,1\}^{n \times m}$ for data, a positive integer k, a threshold value $\tau \in]0,1]$, and real-valued weights w^+ and w^-.

Output: Matrices $\mathbf{B} \in \{0,1\}^{k \times m}$ and $\mathbf{S} \in \{0,1\}^{n \times k}$.

1: **function** ASSOITER $(\mathbf{C}, k, \tau, w^+, w^-)$
2: $\{\mathbf{B}, \mathbf{S}\} \leftarrow$ ASSO$(\mathbf{C}, k, \tau, w^+, w^-)$
3: **repeat**
4: **for** $l = 1, \ldots, k$ **do**
5: $\mathbf{s}_{\cdot l} \leftarrow \arg\max_{\mathbf{s}_{\cdot l} \in \{0,1\}^n} cover(\mathbf{B}, \mathbf{S}, \mathbf{C}, 1, 1)$
6: **until** error $|\mathbf{C} - \mathbf{S} \circ \mathbf{B}|$ does not decrease
7: **return** $\mathbf{B}$ and $\mathbf{S}$

TABLE 1
Details on Generated Data Sets; $\mathbf{D} = \mathbf{S} \circ \mathbf{B}$

dataset	rows of $\mathbf{S}$	columns of $\mathbf{B}$	rows of $\mathbf{B}$	avg. # of 1's/row of $\mathbf{S}$	# of 1's/row of $\mathbf{B}$	noise (%)
set 1	1 000	500	12	4	50	5–40
set 2	1 000	500	12	4, 8	25–200	0
set 3	1 000–16 000	1 000–16 000	10–160	5–80	500–8 000	0

TABLE 2
Real-World Data Sets

						average # of 1's	
dataset	description	rows	columns	# of 1's in data	density (%)	per row	per column
NSF Abstracts	sample	12 841	4 894	564 462	0.9	43.96	115.34
20 Newsgroups	full	19 997	5 163	915 707	0.9	45.79	177.36
	sample	10 000	5 163	455 526	0.9	45.55	88.23
Digits	full	2 000	240	291 654	60.8	145.83	1 215.23
Courses	full	2 405	615	52 739	3.6	21.92	85.75
Paleo	full	124	139	1 978	11.5	15.95	14.23
VotesYes	full	773	196	99 924	66.0	129.27	509.82
VotesNo	full	773	196	31 740	21.0	41.06	161.94

7 Experimental Results

We have performed tests using Algorithm 1 and its variations (i.e., Algorithms 2–5) on generated and real-world data sets. The goals of the experiments are to verify whether ASSO produces intuitive basis vectors, to check whether ASSO can reconstruct the basis vectors used to generate artificial data, and to compare the reconstruction accuracy of ASSO and its variations against other well-known methods for both real and generated data.

This section is organized as follows: Section 7.1 presents the data and error measures used, Section 7.2 explains the methods used, and Section 7.3 gives the results.

7.1 Data and Error Measures

We used both generated and real-world data sets in our experiments. With the generated data sets, we were able to concentrate on a few characteristics of the data, while the real-world data sets represent a wide variety of different Boolean data sets, ranging from small and dense to large and sparse data sets.

7.1.1 Generated Data

We generated three sets of data to test the effects of 1) noise, 2) overlap between basis vectors, and 3) input size. First, a set of basis vectors was generated. Then, random subsets of these basis vectors were used to generate the data rows. Finally, random uniform noise was added. Details on the parameters used to generate the three sets of data are shown in Table 1.

7.1.2 Real Data

The real data consist of the following data sets: NSF Abstracts, 20 Newsgroups, Digits, Courses, Paleo, and Votes. The basic properties of the data sets are given in Table 2.

NSF Abstracts[1] contains document-word information on a collection of project abstracts submitted for funding by NSF. 20 Newsgroups[2] is a collection of approximately 20,000 newsgroup documents across 20 different newsgroups [40]. Digits[3] is a collection of 1,000 binary images of handwritten digits [41]. Paleo[4] contains information of species' fossils found in specific paleontological sites in Europe [42]. Courses is a student-course data set of courses completed by the CS students of the University of Helsinki. Votes[5] is a plenary voting data in the Finnish Parliament during years 1999–2001 [43]. It contains voting behaviors in the plenary votes for each Member of Parliament (MP).

NSF Abstracts, 20 Newsgroups, and Votes were preprocessed before using them. The words in NSF Abstracts and 20 Newsgroups were at first stemmed using Porter stemmer. For NSF Abstracts, we made a random sample of the data. We then removed the least and the most frequent stemmed words. For NSF Abstracts, we removed all words that occurred in less than 10 (sampled) abstracts or in more than 9,999 (sampled) abstracts; the figures for 20 Newsgroups were 36 and 9,999, respectively. A random sample of 20 Newsgroups was used for comparisons between different algorithms due to memory constraints of SVD and NMF implementations; to study the quality of the basis vectors, we used the full data. In Votes data set, we removed all MPs having unknown voting behaviors (due to not being an MP the whole time 1999-2001). In addition, since an MP can cast four different types of votes, namely, "Yea," "Nay," "Abstain," and "Absent," we made two data sets: VotesYes identifies "Yea" as a 1 and everything else as a 0, and VotesNo identifies "Nay" as a 1 and everything else as a 0.

1. http://kdd.ics.uci.edu/databases/nsfabs/nsfawards.html.

2. http://people.csail.mit.edu/jrennie/20Newsgroups/.

3. http://archive.ics.uci.edu/ml/machine-learning-databases/mfeat/.

4. NOW public release 030717, available from http://www.helsinki.fi/science/now/.

5. http://www.fsd.uta.fi/english/data/catalogue/FSD2117/meF2117e.html.

7.1.3 Error Measures

We used two measures to quantify the error of the approximation: sum-of-absolute-values distance d_1, defined as

$$d_1(\mathbf{A}, \mathbf{B}) = \sum_i \sum_j |a_{ij} - b_{ij}|,$$

and Frobenius distance d_2, defined as

$$d_2(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_i \sum_j (a_{ij} - b_{ij})^2}.$$

Note that if both $\mathbf{A}$ and $\mathbf{B}$ are Boolean, then Frobenius distance is the square root of d_1 distance, i.e., $d_2(\mathbf{A}, \mathbf{B}) = \sqrt{d_1(\mathbf{A}, \mathbf{B})}$. Thus, d_1 is our primary distance function.

7.2 Methods

The algorithm for DBP was that from Section 6, i.e., Algorithm 1. For synthetic data, weights w^+ and w^- were both set to 1 (i.e., no weighting), and the threshold value τ was selected to be three percentage units smaller than the actual noise. For real data sets, several different values for τ and weights were tried and the best combination was selected. Selecting $w^+ = w^- = 1$ usually gave the least-error answer, but on few cases, best answer was given by $w^+ = 2$. However, we find the weights useful in order to get more intuitive results, as reported in Section 7.3.4.

Similar selection of the parameters was also used for the variations of ASSO, that is, for ASSO + trans, ASSO + clust, ASSO + opt, and ASSO + iter. For the clustering algorithm in ASSO + clust, we used k-Means algorithm from SOM Toolbox.[6] The core of all these methods, the ASSO algorithm, was implemented using C in GNU/Linux operating system. In addition, ASSO + opt and ASSO + iter are purely C programs. The other methods, ASSO + trans and ASSO + clust, were implemented using Matlab for data preprocessing step.[7]

We did not use ASSO + trans or ASSO + clust for synthetic data sets, as both of these methods assume such properties of the data that we know were not existent in the data. In addition, we did not use ASSO + opt for real data sets, as its exponential computational time would have easily become infeasible with larger data sets and more basis vectors.

The results of the aforementioned algorithms were compared against the results given by SVD, NMF, LPCA, and their variations. None of these methods solves DBP exactly as the resulting decompositions are real-valued. Thus, the results from these methods are not directly comparable to the results from our methods. In order to ease the comparison, we also used some variations of SVD and NMF. The NMF algorithm we used was described in [44]. All of these methods were implemented using Matlab.

Rounded SVD and rounded NMF are like SVD and NMF, but before computing the reconstruction error, we rounded the representation matrix to a Boolean one. That is, the factor matrices were still real-valued, but we rounded their product. The rounding was done in a simple fashion by setting all values less than 0.5 to 0 and all others to 1.

Rounded versions of SVD and NMF still have real-valued factor matrices. As DBP requires Boolean factor matrices, we also used variations of SVD and NMF, where the factor matrices were rounded and the representation was constructed using Boolean matrix multiplication. We call these methods 0-1 SVD and 0-1 NMF. Here, we rounded the factor matrices by trying many different thresholds, separately for each factor matrix, and selecting those thresholds that gave the lowest error. These methods were used only for real data sets.

The results reported for the LPCA are somewhat different from the other results because LPCA returns a probability distribution. The results reported for LPCA are obtained by first running the LPCA algorithm 13 times and then selecting the distribution with the highest log-likelihood. The result LPCA gave was a distribution matrix $\mathbf{P}$, where $p_{ij} \in [0, 1]$. To obtain a binary matrix, we rounded $\mathbf{P}$ from 0.5 (i.e., normal rounding). Rounding was selected as it gave smaller reconstruction error than the expected or empirical (sampling from distribution) error.

We also give a brief study of the performance of the DBPP algorithm. For that, we used the k-median algorithm by Arya et al. [38] with one simultaneous local swap.

7.3 Results

7.3.1 Reconstructing the Basis Vectors from Generated Data

We studied the effects of noise and overlap between basis vectors to the reconstruction error. The main measure was the d_1 distance.

The effects of noise are illustrated in Fig. 1a. Lines for plain SVD and NMF coincide at the top of the figure, partly because of the logarithmic scale. Likewise, the lines for ASSO + opt and ASSO + iter coincide, i.e., ASSO + iter converged to global optima almost every time. Rounded SVD is the best of the other methods when data has noise, being the best of all methods from 20 percent of noise onward. Rounded NMF has a peak in 10 percent of noise because it failed to converge even near to a global optimum with one data set. The quality of LPCA's result has a dramatic change between nonnoised and noised data sets, but increasing the noise level reduces the quality only moderately.

Fig. 1b illustrates the effects of basis vectors' overlap. The expected overlap of two random vectors is uniquely defined by the number of 1s in basis vectors, i.e., the values in x-axis in Fig. 1b. If basis vectors have high overlap, it becomes harder to distinguish the basis vectors using association confidences. Thus, a higher overlap degrades the quality of ASSO results, as one can clearly see from Fig. 1b. For ASSO + opt, the curve is not so sharp with small values, and indeed, ASSO + opt is the second best method in all cases except the last, again sharing its place with ASSO + iter. The increase of the overlap seems to have similar, yet not so strong, effects to the rounded SVD and NMF, too. When the overlap increases, rounded SVD and NMF have better reconstruction accuracy than the DBP algorithms; they are better than even ASSO + opt in the last

6. Available from http://www.cis.hut.fi/projects/somtoolbox/.

7. Implementations are freely available from the corresponding author's homepage at http://www.cs.helsinki.fi/Pauli.Miettinen/.

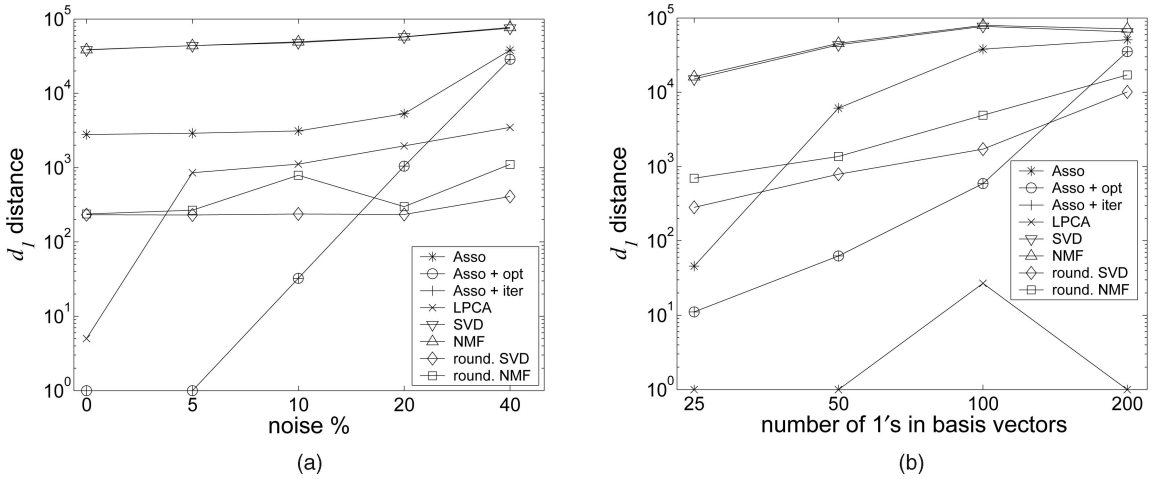

Fig. 1. Reconstruction errors using d_1 as a function of (a) noise (data set 1) and (b) 1s in basis vector (data set 2). Points in plots represent the mean error over 20 random data matrices with the same attributes. Logarithmic scale on y-axis of both plots, and values less than 1 are displayed as 1.

TABLE 3
Reconstruction Error of Real-World Data Sets Using d_1 Distance

			algorithm							
dataset	k	scale	SVD	NMF	r. SVD	r. NMF	0–1 SVD	0–1 NMF	LPCA	ASSO
NSF Abstr.	5	10^6	1.124	1.089	0.563	0.563	5.171	3.328	0.561	0.559
NSF Abstr.	10		1.152	1.091	0.561	0.561	6.065	5.890	0.552	0.554
NSF Abstr.	20		1.197	1.099	0.555	0.556	9.605	10.228	0.530	0.545
20 Newsgr.	5	10^6	0.900	0.875	0.450	0.450	4.185	2.612	0.450	0.451
20 Newsgr.	10		0.928	0.881	0.446	0.447	4.950	4.447	0.440	0.448
20 Newsgr.	20		0.969	0.889	0.440	0.441	7.293	8.096	4.835	0.444
Digits	5	10^5	1.308	1.382	0.763	0.855	1.678	1.113	0.722	1.246
Digits	10		1.070	1.206	0.471	0.610	1.817	0.967	0.350	1.085
Digits	20		0.878	1.028	0.254	0.444	1.678	0.815	0.046	0.878
Courses	5	10^4	6.467	6.204	3.215	3.350	8.202	6.418	3.042	3.749
Courses	10		6.164	5.779	2.770	2.951	14.051	9.840	2.052	3.462
Courses	20		5.949	5.186	2.219	2.495	20.490	17.021	0.642	3.114
Paleo	5	10^3	2.390	2.282	1.379	1.468	1.953	1.632	0.895	1.573
Paleo	10		2.275	2.027	1.020	1.188	2.554	1.504	0.033	1.402
Paleo	20		2.024	1.743	0.515	0.876	3.262	1.659	0.000	1.196
VotesYes	5	10^4	3.015	3.109	1.734	1.900	3.404	2.275	1.584	2.343
VotesYes	10		2.607	2.892	1.381	1.610	2.546	2.136	1.074	2.270
VotesYes	20		2.262	2.589	0.982	1.356	5.072	1.873	0.387	2.047
VotesNo	5	10^4	1.663	1.643	0.918	0.931	2.340	1.153	0.789	1.105
VotesNo	10		1.502	1.497	0.713	0.810	3.129	1.112	0.424	1.007
VotesNo	20		1.361	1.366	0.456	0.634	3.535	1.126	0.005	0.915

case. On the other hand, LPCA shows very different behavior: its results differ from the optimum only in the case of 100 1s in basis vectors, thus making it the best method in every case.

In both figures, ASSO + opt seems to start with considerably better results than the vanilla ASSO algorithm, but this difference is almost lost in the last points of both plots. This indicates that while ASSO can find good basis vectors, it has problems in using them. As the quality of basis vectors decrease (due to the noise or overlapping), the exponential-time optimization looses its significance. Furthermore, ASSO + iter seems to be able to avoid local optima, attaining the quality of ASSO + opt in almost every case.

7.3.2 Reconstruction Errors for Real Data

Reconstruction errors for the real data sets are given in three tables. Tables 3 and 4 compare the vanilla ASSO algorithm against other methods using d_1 and d_2 distances, respectively. Table 5 compares different variations of the ASSO algorithm, namely, ASSO + trans, ASSO + clust, and ASSO + iter, against the plain ASSO algorithm. Table 5 does not contain results with NSF Abstracts or 20 Newsgroups as some of the variations were not able to handle such big data sets. Note also that all algorithms in Table 5 return binary data, i.e., in that case d_2 distance is the square root of d_1. Thus, only d_1 distance is reported.

Overall, LPCA is the best method with real data. Excluding the big and sparse data sets (NSF Abstracts and 20 Newsgroups), it is always the best method with d_1 distance, and often the best even with the d_2 distance. With NSF Abstracts and 20 Newsgroups, the results are not so clear; for example, Table 3 shows that in NSF Abstracts with $k = 5$, ASSO is the best method in d_1 distance. The quality of LPCA's results with 20 Newsgroups decrease dramatically when k is increased from 10 to 20. However,

TABLE 4
Rounded Reconstruction Error of Real-World Data Sets Using d_2 Distance

		algorithm							
dataset	k	SVD	NMF	r. SVD	r. NMF	0–1 SVD	0–1 NMF	LPCA	ASSO
NSF Abstr.	5	727	728	751	750	2 274	1 825	750	748
NSF Abstr.	10	719	721	749	749	2 463	2 427	744	745
NSF Abstr.	20	709	713	745	746	3 099	3 198	728	738
20 Newsgr.	5	649	650	671	672	2 046	1 616	671	672
20 Newsgr.	10	643	644	668	669	2 225	2 109	664	670
20 Newsgr.	20	634	637	664	665	2 701	2 845	2 199	666
Digits	5	239	248	276	293	410	334	269	353
Digits	10	201	221	217	247	426	311	187	329
Digits	20	168	196	159	211	410	286	68	296
Courses	5	165	167	179	183	286	253	174	194
Courses	10	154	158	166	172	375	314	143	186
Courses	20	141	147	149	158	453	413	80	176
Paleo	5	32	32	37	38	44	40	30	40
Paleo	10	28	30	32	34	51	39	6	37
Paleo	20	24	26	23	30	57	41	0	35
VotesYes	5	116	119	132	138	185	151	126	153
VotesYes	10	104	112	118	127	160	146	104	151
VotesYes	20	90	102	99	116	225	137	62	143
VotesNo	5	84	86	96	97	153	107	89	105
VotesNo	15	69	75	75	84	120	106	65	100
VotesNo	20	64	72	68	80	188	106	7	96

this phenomenon could be caused by some issues that are not directly related to the LPCA algorithm (numerical instability, for example).

While ASSO is not always a match for SVD or LPCA, it is always better than the other two methods that have Boolean factor matrices, namely, 0-1 SVD and 0-1 NMF.

Results in Table 5 show that ASSO + iter can always improve the result given by the plain ASSO algorithm. The results from the other variations, ASSO + trans and ASSO + clust, do not have any simple relation with the results of ASSO. This agrees with the intuition that the usefulness of ASSO + trans and ASSO + clust is highly data-dependent.

7.3.3 Empirical Time Complexity

Set 3 was used to verify the empirical time complexity of the algorithm. We only studied the empirical time complexity of the plain ASSO algorithm, as it is the backbone of the other methods. The results obtained agreed with the theoretical complexity results perfectly, i.e., the running time of Algorithm 1 increased linearly with the number of rows in data and with the size of the basis, and quadratically with the number of columns in data.

7.3.4 Quality of Basis Vectors for Real Data

We used the NSF Abstracts and 20 Newsgroups data sets to examine the quality of the ASSO basis vectors. As the words in the data were stemmed, so were the words in the results. To increase the readability, we have returned the words to their original singular form whenever that form has been clear. For some basis vectors, we only report a subset of words. Three dots at the end of a basis vector indicate this.

For NSF Abstracts, we used $\tau = 0.3$ and $w^+ = 6$ as the set of parameters that gave the most intuitive results. Examples of basis vectors and representative words are given as follows:

1. fund, NSF, year;
2. cell, gene, molecular, protein, ...;
3. gopher, Internet, network, world, wide, web, ...;
4. behavior, effect, estim, impact, measure, model, overestimate, predict, test, ...;
5. course, education, enroll, faculty, institute, school, student, undergraduate, ...; and
6. abstract, don, set.

Many of the basis vectors found consisted of words typical to a specific field of science. Examples 2 and 3 are of this type. Some basis vectors were of more general type of words, e.g., words used in science in general (example 4), or words referring to education (example 5). Due to the nature of the data, words related to money and funding were also found in basis vectors (example 1). Despite the fact that most of the resulting basis vectors were very natural, also less natural ones were found. Example 6 illustrates this behavior.

TABLE 5
Differences between Variations of the ASSO Algorithm, d_1 Distance

			ASSO +			
dataset	k	scale	trans	clust	iter	ASSO
Digits	5	10^5	1.376	1.268	1.134	1.246
Digits	10		1.202	1.158	0.973	1.085
Digits	20		1.008	1.036	0.712	0.878
Courses	5	10^4	3.758	4.028	3.695	3.749
Courses	10		3.475	3.862	3.376	3.462
Courses	20		3.198	3.647	3.052	3.114
Paleo	5	10^3	1.573	1.641	1.569	1.573
Paleo	10		1.394	1.540	1.392	1.402
Paleo	20		1.185	1.539	1.177	1.196
VotesYes	5	10^4	2.427	2.462	2.254	2.343
VotesYes	10		2.386	2.207	2.119	2.270
VotesYes	20		2.288	2.182	1.759	2.047
VotesNo	5	10^4	1.068	1.113	1.089	1.105
VotesNo	10		0.992	1.098	0.955	1.007
VotesNo	20		0.928	1.054	0.859	0.915

For 20 Newsgroups, no specific pair of parameters was better than the other, as most of the results were very intuitive. The examples given here are obtained setting $\tau = 0.4$ and $w^+ = 4$, leading to somewhat wordy basis vectors. Example basis vectors are:

1. disk, FAQ, FTP, newsgroup, server, UNIX, Usenet, ...;
2. bible, Christ, church, faith, Jesus, Lord, Paul, ...;
3. playoff, team, win;
4. agent, BATF, cult, FBI, fire, Koresh, Waco, ...;
5. Arab, Israel, Jew, land, Palestinian, war, ...;
6. American, announce, campaign, crime, federal, fund, office, policy, president, ...;
7. Armenia, Azerbaijan, border, defend, military, soviet, war, ...; and
8. earth, orbit, space.

The first example is clearly computer-related, second contains religious words, specific for Christianity, and third is about sports. Finding such basis vectors could hardly be regarded as a surprise, as the 20 Newsgroups data contains many newsgroups in each of these subjects.

To understand the fourth example, one needs to have some knowledge of the recent history of the United States. All words are related to the so-called Waco siege in 1993, where FBI and BATF raided a religious group led by David Koresh in Waco, Texas.

Examples 5 and 7 are partly overlapping; word "war" appears in both of them. However, the wars are different ones. Example 5 clearly points to Israel-Palestinian conflicts, while example 7 is about Armenia and Azerbaijan, neighbors and former soviet republics.

Example 6 includes words usually seen in political contexts while the last example is from a space-related newsgroup.

In summary, our algorithm was able to find basis vectors that were intuitive and informative. For example, there was no prior information to tell that the Waco siege was discussed in some newsgroup. Yet, it came out very strongly in the results, indicating that it had been a "hot topic" for some time, at least. Among the informative and intuitive basis vectors, there were also less intuitive and less informative ones. Careful selection of algorithm's parameters helps reduce their number, but they cannot be totally avoided. However, we do not find that as a problem, as there are always meaningful and interesting basis vectors present in the answers, too.

7.3.5 The DBPP Algorithm

Finally, we briefly report results on applying an algorithm for DBPP. Recall from Section 5.2 that we can solve DBPP using any existing algorithm for BKMP. Here, we used Arya et al.'s local-search heuristic [38]. As the algorithm is well known and as the results are not directly comparable to other results in this section, we only report results from a single domain, namely, 20 Newsgroups.

The reconstruction errors (in d_1 distance) were 497,768, 494,511, and 489,562 for k equal to 5, 10, and 20, respectively. Comparing these values to those in Table 3 shows that the ASSO algorithm can produce better results. This is not surprising, as DBPP has more restrictions on feasible solutions than DBP. Yet, the results show that ASSO is able to take advantage on this freedom.

The empirical quality of the basis vectors produced by Arya et al.'s algorithm was not that good, either. Most basis vectors contained many words from distinct fields making them hard to interpret. Not all of the results were completely unintuitive, though. A basis vector containing words "George," "Bush," "Jimmy," "Carter," "president," "Arab," "Jew," and "Israel" was an example of a good answer.

8 Discussion and Conclusions

We have described the DBP, investigated its computational complexity, given a simple algorithm for it, and shown empirical results on the behavior of the algorithm. The results indicate that the algorithm discovers intuitively useful basis vectors. In generated data, the method can reconstruct the basis vectors that were used to generate the data; this holds even with high amounts of noise. While the normal ASSO algorithm has some problems on using the basis vectors, simple iterative update can greatly improve the results.

Yet, in many cases, SVD has lower reconstruction error than ASSO (or even ASSO + iter). There are several possible reasons for this. The first possibility is that SVD is in some sense inherently more powerful than DBP. This is of course vaguely expressed. While we know that SVD is optimal with respect to the Frobenius norm, we also know that the Boolean rank of a matrix can be much smaller than its real rank. On one hand, SVD in some ways has more power than DBP, as SVD works on the continuous values; on the other hand, DBP can take advantage of the Boolean semiring on which it operates. This suggests that the relative performance of DBP algorithms against SVD should improve as the overlap between basis vectors increases.

The second alternative reason for the good performance of SVD is that the ASSO algorithm is suboptimal. This suboptimality certainly degrades the results: For example, overlap between the basis vectors makes them harder to be discovered, making the algorithm unable to take advantage of the possibly smaller Boolean rank. However, for our generated data, in many cases, the ASSO algorithm reconstructs the original basis vectors perfectly. Thus, at least for those data sets the algorithm is sufficiently good. The results from ASSO + opt indicate that using the basis vectors is a major source of difficulties for the ASSO algorithm: it does find correct basis vectors, but it is unable to use them correctly.

We have shown that Boolean approaches to matrix decomposition form a viable alternative for traditional methods. For further work, it would be of interest to understand the relationship between the approximate Boolean and real ranks of binary matrices better. In addition, investigating the complexity of using basis vectors and improving the algorithm in that respect would be useful.

Acknowledgments

The authors are grateful to Ella Bingham, Jouni Seppänen, and Ata Kabán for providing us with the Matlab implementations of NMF and LPCA, and to Aleks Jakulin for informing us about the Votes data set. A preliminary verision of this paper appeared in PKDD '06 [1].

REFERENCES

[1] P. Miettinen, T. Mielikäinen, A. Gionis, G. Das, and H. Mannila, "The Discrete Basis Problem," *Proc. 10th European Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD '06)*, pp. 335-346, 2006.

[2] G. Golub and C. van Loan, *Matrix Computations.* Johns Hopkins Univ. Press, 1996.

[3] D. Lee and H. Seung, "Learning the Parts of Objects by Non-Negative Matrix Factorization," *Nature*, vol. 401, pp. 788-791, 1999.

[4] D. Blei, A. Ng, and M. Jordan, "Latent Dirichlet Allocation," *J. Machine Learning Research*, vol. 3, pp. 993-1022, 2003.

[5] W. Buntine, "Variational Extensions to EM and Multinomial PCA," *Proc. 13th European Conf. Machine Learning (ECML '02)*, pp. 23-34, Aug. 2002.

[6] P. Paatero and U. Tapper, "Positive Matrix Factorization: A Non-Negative Factor Model with Optimal Utilization of Error Estimates of Data Values," *Environmetrics*, vol. 5, pp. 111-126, 1994.

[7] J.E. Cohen and U.G. Rothblum, "Nonnegative Ranks, Decompositions, and Factorizations of Nonnegative Matrices," *Linear Algebra and Its Applications*, vol. 190, pp. 149-168, 1993.

[8] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, and R.J. Plemmons, "Algorithms and Applications for Approximate Nonnegative Matrix Factorization," *Computational Statistics and Data Analysis*, vol. 52, pp. 155-173, 2007.

[9] T. Hofmann, "Probabilistic Latent Semantic Indexing," *Proc. 22nd Ann. Int'l ACM Conf. Research and Development in Information Retrieval (SIGIR '99)*, pp. 50-57, Aug. 1999.

[10] W. Buntine and A. Jakulin, "Discrete Component Analysis," *Proc. Subspace, Latent Structure and Feature Selection, Statistical and Optimization, Perspectives Workshop (SLSFS '05)*, pp. 1-33, 2006.

[11] E. Bingham, A. Kabán, and M. Fortelius, "The Aspect Bernoulli Model: Multiple Causes of Presences and Absences," to be published in *Pattern Analysis and Applications*, 2008.

[12] J. Seppänen, E. Bingham, and H. Mannila, "A Simple Algorithm for Topic Identification in 0-1 Data," *Proc. Seventh European Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD '03)*, pp. 423-434, 2003.

[13] A.I. Schein, L.K. Saul, and L.H. Ungar, "A Generalized Linear Model for Principal Component Analysis of Binary Data," *Proc. Ninth Int'l Workshop Artificial Intelligence and Statistics (AI & Statistics)*, 2003.

[14] D.P. O'Leary and S. Peleg, "Digital Image Compression by Outer Product Expansion," *IEEE Trans. Comm.*, vol. 31, no. 3, pp. 441-444, 1983.

[15] T.G. Kolda and D.P. O'Leary, "A Semidiscrete Matrix Decomposition for Latent Semantic Indexing in Information Retrieval," *ACM Trans. Information Systems*, vol. 16, no. 4, pp. 322-346, 1998.

[16] M.W. Berry, S.A. Pulatova, and G.W. Stewart, "Algorithm 844: Computing Sparce Reduced-Rank Approximations to Sparce Matrices," *ACM Trans. Math. Software*, vol. 31, no. 2, pp. 252-269, 2005.

[17] P. Drineas, R. Kannan, and M.W. Mahoney, "Fast Monte Carlo Algorithms for Matrices III: Computing a Compressed Approximate Matrix Decomposition," *SIAM J. Computing*, vol. 36, no. 1, pp. 184-206, 2006.

[18] P. Drineas, M.W. Mahoney, and S. Muthukrishnan, *Relative-Error CUR Matrix Decompositions*, arXiv:0708.3696v1 [cs.DS], http://arxiv.org/abs/0708.3696, Aug. 2007.

[19] M. Koyutürk, A. Grama, and N. Ramakrsihnan, "Compression, Clustering, and Pattern Discovery in Very-High-Dimensional Discrete-Attribute Data Sets," *IEEE Trans. Knowledge Data Eng.*, vol. 17, pp. 447-461, 2005.

[20] A. Gionis, H. Mannila, and J.K. Seppänen, "Geometric and Combinatorial Tiles in 0-1 Data," *Proc. Eighth European Conf. Principles and Practice of Knowledge Discovery in Databases (PKDD '04)*, pp. 173-184, 2004.

[21] F. Geerts, B. Goethals, and T. Mielikäinen, "Tiling Databases," *Proc. Seventh Int'l Conf. Discovery Science (DS '04)*, pp. 278-289, 2004.

[22] J. Besson, R. Pensa, C. Robardet, and J.-F. Boulicaut, "Constraint-Based Mining of Fault-Tolerant Patterns from Boolean Data," *Proc. Fourth Int'l Workshop Knowledge Discovery in Inductive Databases (KDID '06)*, pp. 55-71, 2006.

[23] N. Mishra, D. Ron, and R. Swaminathan, "A New Conceptual Clustering Framework," *Machine Learning*, vol. 56, pp. 115-151, 2004.

[24] R.K. Brayton, G.D. Hachtel, and A.L. Sangiovanni-Vincentelli, "Multilevel Logic Synthesis," *Proc. IEEE*, vol. 78, no. 2, pp. 264-300, 1990.

[25] J.A. Hartigan, "Direct Clustering of a Data Matrix," *J. Am. Statistical Assoc.*, vol. 67, no. 337, pp. 123-129, 1972.

[26] A. Banerjee, I.S. Dhillon, J. Ghosh, S. Merugu, and D.S. Modha, "A Generalized Maximum Entropy Approach to Bregman Co-Clustering and Matrix Approximations," *Proc. 10th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '04)*, pp. 509-514, 2004.

[27] S. Madeira and A. Oliveira, "Biclustering Algorithms for Biological Data Analysis: A Survey," *IEEE/ACM Trans. Computational Biology and Bioinformatics*, vol. 1, no. 1, pp. 24-45, Jan.-Mar. 2004.

[28] C. Robardet and F. Feschet, "Efficient Local Search in Conceptual Clustering," *Proc. Fourth Int'l Conf. Discovery Science (DS '01)*, pp. 323-335, 2001.

[29] J. Vaidya, V. Atluri, and Q. Guo, "The Role Mining Problem: Finding a Minimal Descriptive Set of Roles," *Proc. ACM Symp. Access Control Models and Technologies (SACMAT '07)*, pp. 175-184, 2007.

[30] H. Lu, J. Vaidya, and V. Atluri, "Optimal Boolean Matrix Decomposition: Application to Role Engineering," *Proc. IEEE Int'l Conf. Data Eng. (ICDE '08)*, pp. 297-306, Apr. 2008.

[31] S.D. Monson, N.J. Pullman, and R. Rees, "A Survey of Clique and Biclique Coverings and Factorizations of (0, 1)-Matrices," *Bull. Inst. Combinatorics and Its Applications*, vol. 14, pp. 17-86, 1995.

[32] D.A. Gregory and N.J. Pullman, "Semiring Rank: Boolean Rank and Nonnegative Rank Factorizations," *J. Combinatorics, Information and System Sciences*, vol. 8, no. 3, pp. 223-233, 1983.

[33] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, 1979.

[34] H.U. Simon, "On Approximate Solutions for Combinatorial Optimization Problems," *SIAM J. Discrete Math.*, vol. 3, no. 2, pp. 294-310, 1990.

[35] R.G. Downey and M.R. Fellows, "Parameterized Complexity," *Monographs in Computer Science.* Springer-Verlag, 1999.

[36] J. Flum and M. Grohe, *Parameterized Complexity Theory.* Springer, 2006.

[37] N. Megiddo and K. Supowit, "On the Complexity of Some Common Geometric Location Problems," *SIAM J. Computing*, vol. 13, no. 1, pp. 182-196, 1984.

[38] V. Arya, N. Garg, R. Kjandekar, A. Meyerson, K. Munagala, and V. Pandit, "Local Search Heuristics for *k*-Median and Facility Location Problems," *SIAM J. Computing*, vol. 33, no. 3, pp. 544-562, 2004.

[39] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proc. ACM SIGMOD '93*, pp. 207-216, May 1993.

[40] K. Lang, "Newsweeder: Learning to Filter Netnews," *Proc. 12th Int'l Conf. Machine Learning (ICML '95)*, pp. 331-339, 1995.

[41] D. Newman, S. Hettich, C. Blake, and C. Merz, "UCI Repository of Machine Learning Databases," http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.

[42] M. Fortelius, *Neogene of the Old World Database of Fossil Mammals (NOW '05)*, http://www.helsinki.fi/science/now/, 2005.

[43] A. Pajala and A. Jakulin, "Plenary Votes in the Finnish Parliament during 1991-2005," Tampere: Finnish Social Science Data Archive, http://www.fsd.uta.fi/english/, 2006.

[44] D. Lee and H. Seung, "Algorithms for Non-Negative Matrix Factorization," *Advances in Neural Information Processing Systems*, vol. 13, pp. 556-562, 2001.

Pauli Miettinen received the MSc degree in computer science from the University of Helsinki in 2006. He is a graduate student at Helsinki Institute for Information Technology, University of Helsinki. His research interests include data mining, theoretical computer science, and algorithmic data analysis.

Taneli Mielikäinen received the PhD degree from the University of Helsinki in 2005. He is a senior research scientist at the Palo Alto Systems Research Center of Nokia and an adjunct professor of computer science at the University of Helsinki. His research interests include algorithmic data analysis, combinatorial optimization, and computational privacy preservation. He has been serving in the program committees of many international conferences.

Aristides Gionis received the PhD degree from Stanford University in 2003. He is currently a senior research scientist at Yahoo! Research, Barcelona. His research interests include data mining, Web mining, and algorithmic data analysis. He has been serving in the program committees of many international conferences.

Gautam Das received the PhD degree in computer science from the University of Wisconsin, Madison, in 1990. He has been an associate professor in the Computer Science and Engineering Department, University of Texas, Arlington, since 2004. He has held positions at the University of Memphis, Compaq, and most recently at Microsoft Research. In addition, he has held visiting positions at the Max-Planck-Institute for Informatics, Saarbrücken, Germany, University of Helsinki, and the Indian Institute of Science, Bangalore, India. His research interests include data mining and knowledge discovery, databases, algorithms, and computational geometry. His research has been supported by the US National Science Foundation (NSF), US Office of Naval Research (ONR), Microsoft, Cadence Design Systems, and Apollo Data Technologies. He has served in numerous program committees, as a PC cochair in the First International Workshop on Ranking in Databases (DBRank 2007) and Ninth ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD 2004), and as a guest editor for the *ACM Transactions on Knowledge Discovery from Data (TKDD)*.

Heikki Mannila received the PhD degree in computer science from the University of Helsinki in 1985. After some time at the University of Tampere and various researcher positions, in 1989 he was appointed as a professor of computer science at the University of Helsinki. He was a visiting professor at the Technical University of Vienna in 1993 and a visiting researcher at Max Planck Institute for Computer Science, Saarbrücken, Germany, in 1995-1996. He moved to Microsoft Research, Redmond, Washington, in 1998, and then came back to Finland to Nokia Research in 1999, where he stayed until the end of 2001. After that, he was the research director of the basic research unit of Helsinki Institute for Information Technology in 2002-2004. Since 1999, he has been a professor of computer science at Helsinki University of Technology. He is currently an academy professor (2004-2009). His research group is located partly in Helsinki University of Technology and partly in the University of Helsinki. He received the ACM SIGKDD Innovation Award in 2003. He is the author of two books and more than 150 refereed articles in computer science and related areas. The book *Principles of Data Mining*, with David Hand and Padhraic Smyth, is available also in Chinese and in Polish.